

100  
Data Processing System

Figure 1

10/26/2009 9:56:00

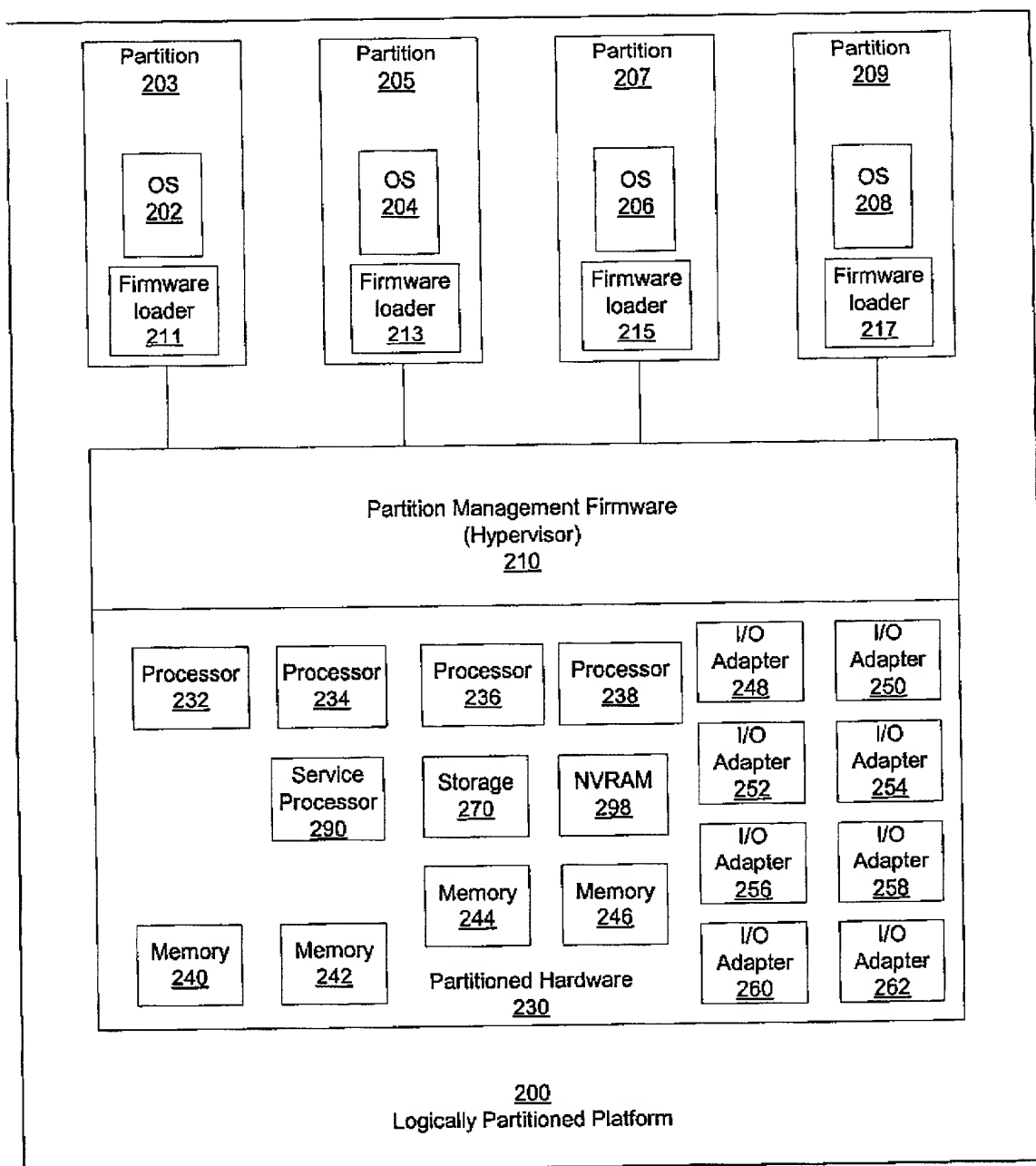
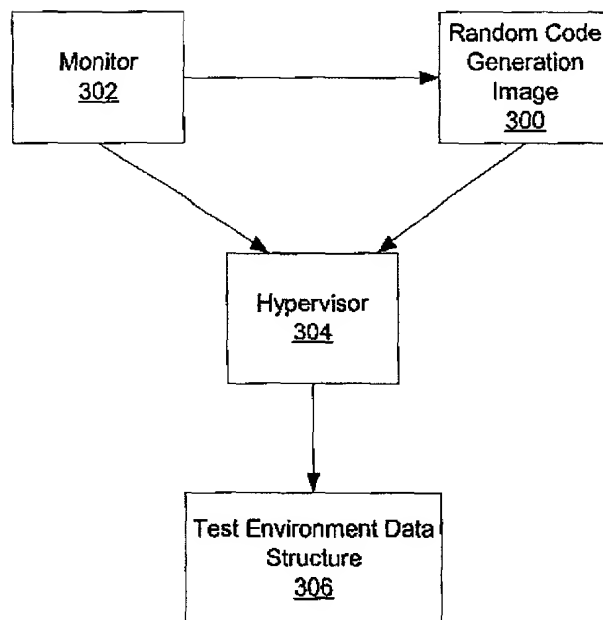


Figure 2

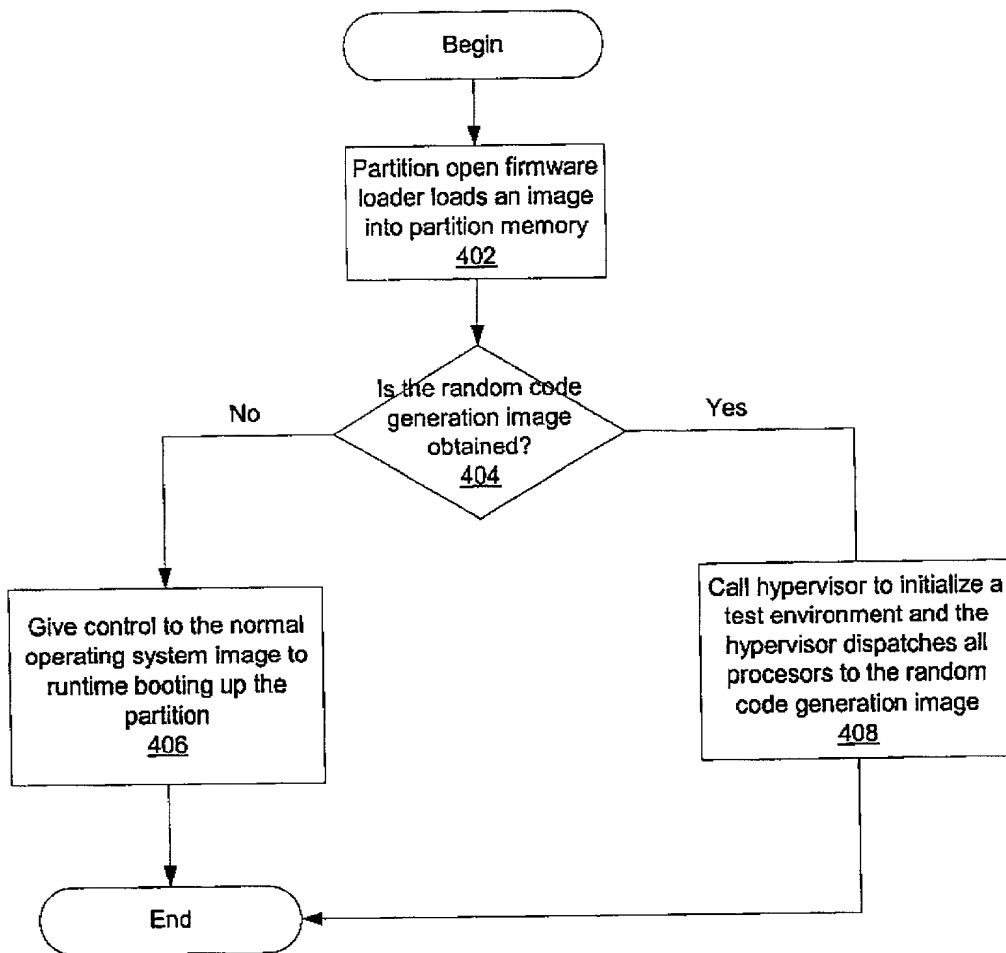
# Figure 3

Lee et al.  
AUS920010672US1  
Method and Apparatus for Verifying Hardware  
Implementation of a Processor Architecture in a  
Logically Partitioned Data Processing System  
Page 3 of 9



# Figure 4

Lee et al.  
AUS920010672US1  
Method and Apparatus for Verifying  
Hardware Implementation of a  
Processor Architecture in a Logically  
Partitioned Data Processing System  
Page 4 of 9



095500-0974  
T02260-0059650

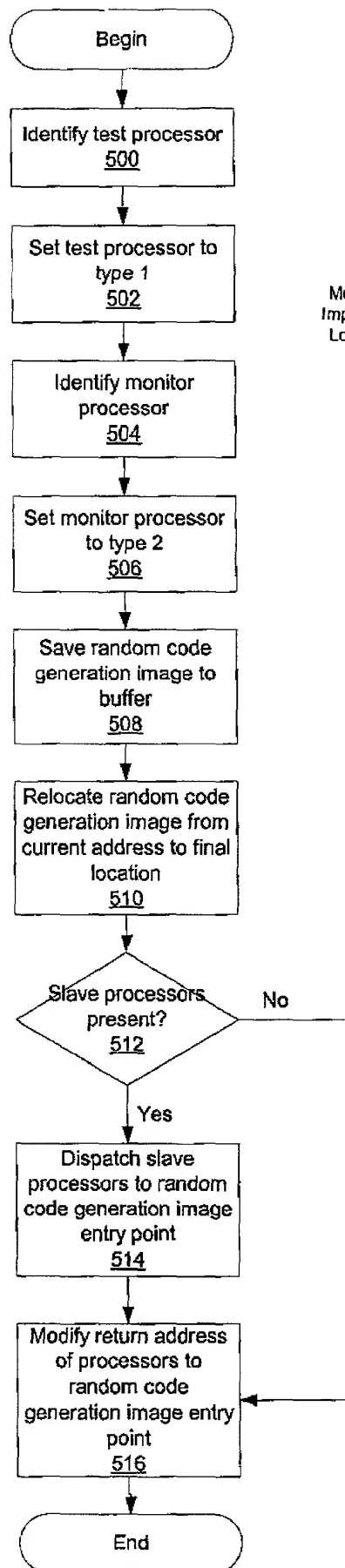
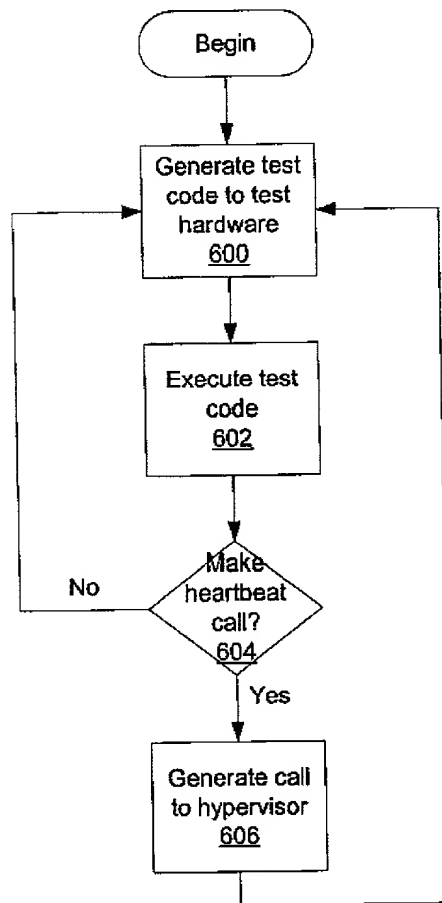


Figure 5

Lee et al.  
AUS920010672US1  
Method and Apparatus for Verifying Hardware  
Implementation of a Processor Architecture in a  
Logically Partitioned Data Processing System  
Page 5 of 9

# Figure 6

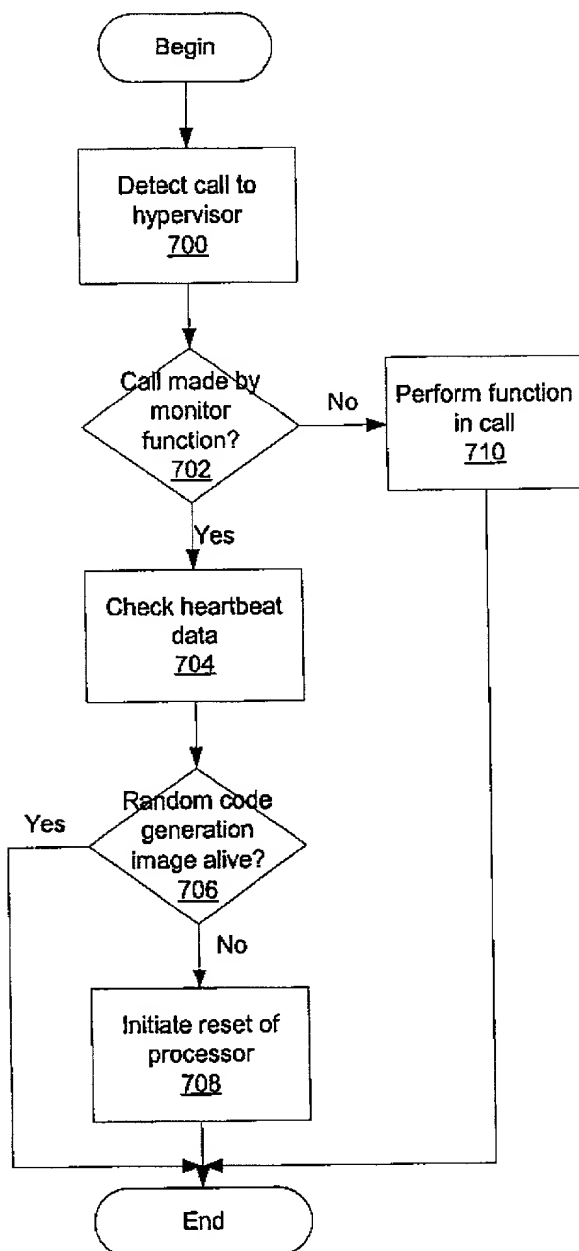
Lee et al.  
AUS920010672US1  
Method and Apparatus for Verifying  
Hardware Implementation of a  
Processor Architecture in a Logically  
Partitioned Data Processing System  
Page 6 of 9



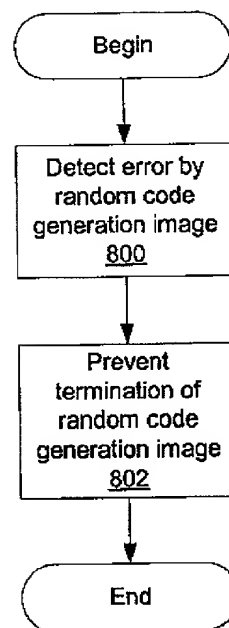
FOI260:00059660

# Figure 7

Lee et al.  
AUS920010672US1  
Method and Apparatus for Verifying  
Hardware Implementation of a  
Processor Architecture in a Logically  
Partitioned Data Processing System  
Page 7 of 9



# Figure 8



## Figure 9

Lee et al.  
AUS920010672US1  
Method and Apparatus for Verifying  
Hardware Implementation of a  
Processor Architecture in a Logically  
Partitioned Data Processing System  
Page 8 of 9

```

struct
{
    unsigned long heartbeat = 0; /* increment one each time by the LTK processor making the
    "heartbeat" h-call */
    unsigned long last_heartbeat = -1; /* capture the heartbeat into this field by the MONITOR processor */
    unsigned long timestamp = -1; /* time stamp when the capture takes place if there is still
    heartbeat ticking */
} heartbeat[MAX_PROCS];

```

## Figure 10

```

h_heartbeat()
{
    my_proc_id = get_proc_id();
    my_part_id = get_part_id();
    if ( my_part_id == GLOBAL_FW ) return; /* can't call from global FW */
    if ( proc_type[ my_proc_id ] != LTK_TYPE ) return; /* processor is not running LTK */
    else
    {
        extern unsigned long refresh_frequency;
        extern PART_INFO part_info[];
        unsigned long newtime;

        ++heartbeat[ my_proc_id ].heartbeat;
        simple_lock( &part_info[ my_part_id ].refresh_lock ); /* serialize access to
        last_refresh_timestamp */
        newtime = get_current_time_ticks();
        if ( (newtime - part_info[ my_part_id ].last_refresh_timestamp) > refresh_frequency )
        {
            /* time is up for a image refresh */
            restore_LTK_img_from_saved_copy();
            part_info[ my_part_id ].last_refresh_timestamp = get_current_time_ticks();
        }
        simple_unlock( &part_info[ my_part_id ].refresh_lock );
    }
}

```

# Figure 11

Lee et al.  
AUS920010672US1  
Method and Apparatus for Verifying  
Hardware Implementation of a  
Processor Architecture in a Logically  
Partitioned Data Processing System  
Page 9 of 9

1100

```
monitor_code()
{
    my_proc_id = get_proc_id();
    my_sister_id = get_sister_id( my_proc_id );
    if ( proc_type[ my_proc_id ] == MONITOR_TYPE )
    {
        if ( heartbeat[ my_sister_id ].last_heartbeat != heartbeat[ my_sister_id ].heartbeat )
        {
            /* there is heartbeat ticking, capture the new heartbeat and record new timestamp */
            heartbeat[ my_sister_id ].last_heartbeat = heartbeat[ my_sister_id ].heartbeat;
            heartbeat[ my_sister_id ].timestamp = get_current_time_ticks();
        }
        else
        {
            extern unsigned long heartbeat_frequency;
            unsigned long newtime;

            /* if there is no heartbeat which exceeds the required heartbeat observed interval */
            /* initiate a reset to the LTK processor monitored by this processor */
            newtime = get_current_time_ticks();
            if ( (newtime - heartbeat[ my_sister_id ].timestamp) > heartbeat_frequency )
                reset_the_sister_LTK_processor();
        }
    }
}
```

# Figure 12

1200

```
monitor_glue_code:    # at a fixed location in hypervisor memory
    blr               # Simply return
    b    monitor_code # Call the actual monitor code
```